📚 Дополнение к ТЗ: Адаптивное распределение тейк-профитов при недостатке маржи

> Цель: Описать поведение бота, когда объём позиции слишком мал для выставления всех тейк-профитов согласно глобальным настройкам.

> Бот не должен пытаться "растянуть" маржу на все уровни, а должен умело консолидировать доли, чтобы сохранить максимально возможное количество исполняемых ордеров.

---

🔧 1. Общая логика

Бот не обязан выставлять все N тейков, если это приведёт к созданию невалидных ордеров (с объёмом ниже минимального).

Вместо этого он:

1. Проверяет каждый тейк, начиная с последнего (`TP4 → TP3 → TP2`),

2. Если объём одного тейка меньше минимально допустимого — он удаляется,

3. Его процентная доля передаётся предыдущему тейку,

4. Если и тот становится слишком малым — вся накопленная доля равномерно распределяется между оставшимися валидными тейками.

> ✅ Цель: Сохранить как можно больше уровней закрытия, но гарантировать, что каждый ордер будет принят биржей.

---

⚙️ 2. Алгоритм работы

```text

1. Получить сигнал с N тейками и их процентным распределением.

2. Получить размер позиции (в монете, например, DOGE).

3. Для каждой пары получить minQty (минимальный объём ордера) через Bitget API.

4. Начать с последнего тейка (TP\_N):

- Рассчитать объём: size = position\_size percentage / 100

- Если size < minQty:

- Удалить тейк

- Добавить его % к предыдущему тейку

- Перейти к следующему (более раннему)

5. Если после добавления доля всё ещё недостаточна:

- Все накопленные % от удалённых тейков

- Равномерно распределить между оставшимися валидными тейками

6. Вернуть обновлённый список тейков

```

---

💡 3. Пример 1: Доли передаются вперёд, тейк спасён

Сценарий:

- Пара: `DOGEUSDT.P`

- Минимальный объём ордера (`minQty`): 5 DOGE

- Размер позиции: 41 DOGE

- Распределение: `[TP1: 50%, TP2: 30%, TP3: 10%, TP4: 10%]`

Расчёт:

| Тейк | Объём (DOGE) | Хватает? | Действие |

|------|--------------|----------|---------|

| TP4 | 41 × 10% = 4.1 | ❌ Нет | Удалить, +10% → TP3 |

| TP3 | 41 × (10%+10%) = 8.2 | ✅ Да | Оставить |

Итоговое распределение:

`[TP1: 50%, TP2: 30%, TP3: 20%]`

✅ Все тейки валидны.

❌ TP4 удалён, но его доля помогла "спасти" TP3.

---

💡 4. Пример 2: Доли распределяются поровну между оставшимися

Сценарий:

- Пара: `DOGEUSDT.P`

- Минимальный объём ордера (`minQty`): 9 DOGE (высокий порог)

- Размер позиции: 41 DOGE

- Распределение: `[TP1: 50%, TP2: 30%, TP3: 10%, TP4: 10%]`

Расчёт:

| Тейк | Объём (DOGE) | Хватает? | Действие |

|------|--------------|----------|---------|

| TP4 | 4.1 | ❌ | Удалить, +10% → TP3 |

| TP3 | 4.1 + 4.1 = 8.2 | ❌ | Удалить, +20% → далее |

Теперь 20% нужно распределить.

Остаются: TP1 (50%) и TP2 (30%).

❌ Нельзя добавить 20% только к TP2 — нарушилась бы логика.

✅ Нужно разделить 20% поровну между оставшимися:

- TP1: 50% + 10% = 60%

- TP2: 30% + 10% = 40%

Итоговое распределение:

`[TP1: 60%, TP2: 40%]`

---

🛠️ 5. Реализация на Java

```java

@Service

public class TakeProfitOptimizer {

public List<TakeProfitLevel> optimize(List<TakeProfitLevel> originalLevels,

double positionSize,

double minQuantity) {

List<TakeProfitLevel> result = new ArrayList<>(originalLevels);

// Идём с конца

for (int i = result.size() - 1; i > 0; i--) {

TakeProfitLevel level = result.get(i);

double qty = positionSize level.getRatio();

if (qty < minQuantity) {

// Переносим долю к предыдущему

result.get(i - 1).addRatio(level.getRatio());

result.remove(i);

}

}

// Проверяем первый уровень (нет предыдущего)

TakeProfitLevel first = result.get(0);

if (positionSize first.getRatio() < minQuantity && result.size() > 1) {

// Невозможно спасти даже первый — распределяем доли поровну

double totalRatio = result.stream().mapToDouble(TakeProfitLevel::getRatio).sum();

double equalRatio = totalRatio / result.size();

return result.stream()

.map(level -> new TakeProfitLevel(equalRatio))

.collect(Collectors.toList());

}

return result;

}

}

// Вспомогательный класс

public class TakeProfitLevel {

private double ratio; // доля от 0 до 1

public TakeProfitLevel(double ratio) {

this.ratio = ratio;

}

public void addRatio(double add) {

this.ratio += add;

}

public double getRatio() {

return ratio;

}

}

```

---

📊 6. Как получить minQuantity с Bitget

Бот должен запросить информацию о паре:

```http

GET /api/v2/mix/market/symbols?productType=USDT-FUTURES&symbol=DOGEUSDT

```

Ответ содержит:

```json

{

"data": {

"minTradeNum": "5", // минимальный объём ордера в DOGE

"priceScale": "4"

}

}

```

> 💡 Рекомендация: Кэшировать `minTradeNum` с периодическим обновлением (раз в час), чтобы не нагружать API.

---

📝 7. Пояснение для исполнителя

| Вопрос | Ответ |

|-------|--------|

| Почему нельзя просто уменьшить все тейки пропорционально? | Это может привести к тому, что все тейки станут слишком малыми. Лучше сохранить несколько больших, чем потерять все. |

| Почему деление именно поровну? | Это наиболее нейтральный и справедливый способ перераспределения, не нарушающий логику первоначального распределения. |

| Что если останется только один тейк? | Он получает 100%. Но лучше отправить админу уведомление: `⚠️ Маленькая позиция: тейки уменьшены до 1.` |

| Можно ли отключить эту логику? | Нет. Это обязательная защита от ошибок исполнения. |

---